Spatiotemporal Multimodal Interaction for Anomaly Detection in Microservices

Ziwei Yang[1(✉)]

[1] China University of Geosciences (Wuhan), Wuhan Hubei 430078, China
yangzw@aka.yeah.net

Abstract. Microservice architectures are widely adopted in modern systems, making anomaly detection essential for maintaining system stability. However, existing approaches either focus on a single modality or adopt simple feature concatenation across metrics, logs, and traces; such strategies fail to capture interactions among modalities. In particular, they tend to overlook spatiotemporal dependencies, making it difficult to comprehensively capture how anomalies evolve in complex systems, which in turn leads to missed detections and false alarms. To address these limitations, this paper proposes STMID (Spatiotemporal Multimodal Interaction for Microservice Anomaly Detection), a spatiotemporal multimodal interaction method for anomaly detection in microservices. The proposed method introduces a service dependency graph to provide a unified representation of multimodal time-series data in microservice systems. It uncovers cross-modal correlations via multimodal fusion. In addition, it models spatiotemporal dependencies to capture anomalous patterns. Anomalies are measured based on the reconstruction error between the original and reconstructed spatiotemporal dependency modeling results, enabling binary classification between normal and abnormal states. Experimental results on two datasets, MSDS and GAIA, show that the proposed method achieves strong overall performance, with an average F1-score of 0.879. It outperforms most baseline methods, with improvements ranging from 3.85% to 42.95%.

Keywords: Microservices, Multimodal fusion, Deep learning, Anomaly detection

1 Introduction

Microservice architectures are widely adopted in modern enterprises because they support the independent deployment and scaling of service units. In a microservice architecture, services are deployed across instances and interact through complex, evolving invocation relationships [1]. Although this flexible architecture improves system scalability, the sheer number of services and the complexity of traces also create major challenges for operations and maintenance. Once a problem occurs in a single instance, it may quickly affect the performance of the entire system and cause substantial economic losses for enterprises. Therefore, timely and accurate anomaly detection is essential for maintaining system stability and enabling rapid business recovery [2].

In microservice systems, anomalies often appear in multiple data modalities at the same time and are usually correlated. As shown in Figure 1, between 16:00 and 22:00, warning messages appear in the logs of instance A, calls from instance A to instance B time out, and CPU utilization becomes abnormal. Existing multimodal anomaly detection methods typically combine features from different modalities through simple concatenation, while overlooking the correlations between them, which limits detection performance. In addition, anomalies in microservice systems tend to evolve, propagate, and accumulate over time along traces, eventually affecting overall system performance. Existing methods either focus only on temporal relationships while ignoring spatial dependencies or model temporal dynamics and service invocation relationships separately, making it difficult to accurately capture how anomalies occur and spread. Moreover, most existing approaches rely on discriminative models that directly classify features and mainly focus on the decision boundary between normal and anomalous samples, rather than explicitly modeling the underlying data distribution. As a result, they struggle to represent diverse anomaly patterns, and their expressive capability is limited in complex multimodal time-series scenarios.

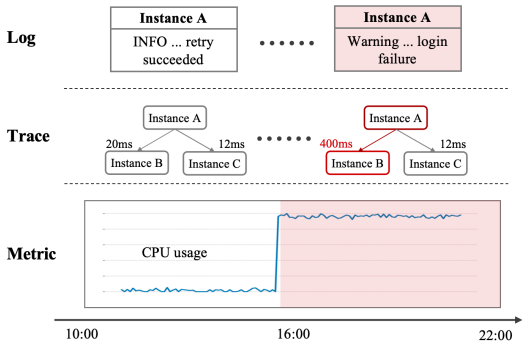


Fig.1. Manifestation of anomalies in multimodal data.

To address these issues, STMID, a semi-supervised multimodal anomaly detection method is proposed. It integrates data preprocessing, cross-modal fusion, and spatiotemporal dependency modeling. In the data preprocessing stage, time-aligned multimodal representations are constructed, and system dependency structures are modeled by traces to characterize anomaly propagation paths. In the cross-modal fusion stage, features from different modalities are mapped into a unified representation space, in which a multi-head attention mechanism is introduced to learn correlations across modalities. In the anomaly detection stage, spatiotemporal dependency modeling is incorporated into a variational autoencoder to capture the combined patterns of multimodal spatiotemporal data, and anomalies are detected based on the reconstruction error.

We evaluate STMID's performance on two open-source datasets collected from microservice systems. Experimental results show that, compared with five state-of-the-art anomaly detection baselines, STMID achieves an average F1-score of 0.879, outperforming most methods with improvements ranging from 3.85% to 42.95%.

In summary, the main contributions of this paper are:

- We propose a cross-modal fusion mechanism that effectively fuses multimodal information by exploring the correlations among different data modalities.
- We design a spatiotemporal dependency modeling method to capture the joint dependency patterns of multimodal data in both temporal evolution and service invocation relationships.
- We propose STMID, a microservice anomaly detection method based on spatiotemporal multimodal interaction. It incorporates a variational autoencoder for generative modeling to learn the distributional characteristics of samples in the latent space, thereby improving the model's ability to detect complex anomaly patterns.

2 Related Work

2.1 Single-modal Anomaly Detection

Single-modal anomaly detection methods typically target a single data source, and model normal patterns to identify anomalies.

Metrics-based methods analyze multivariate time series data. TranAD [3] employs Transformer-based self-attention to capture long-term dependencies. OmniAnomaly [4] models temporal dynamics using GRU combined with a variational autoencoder (VAE). USAD [5] introduces an adversarial dual-autoencoder framework to enhance anomaly discrimination. MAD-GAN [6] leverages generative adversarial networks to model complex temporal distributions, while MTAD-GAT [7] incorporates graph attention networks to capture inter-variable dependencies. MUTANT [8] further integrates temporal GCN with attention-based VAE to improve robustness.

Log-based methods focus on extracting semantic and sequential patterns from logs. DeepLog [9] utilizes LSTM to model event sequences. LogAnomaly [10] enhances semantic representation using template embeddings. LogRobust [11] improves robustness with attention-based Bi-LSTM. MLog [12] introduces semantic-aware encoding with transformer architecture. GAELog [13] integrates graph structures with adversarial autoencoders, while PLELog [14] improves semi-supervised learning through probabilistic label estimation.

Trace-based methods analyze service invocation patterns and latency. TraceAnomaly [15] models trace sequences using deep Bayesian networks. TraceModel [16] constructs service dependency graphs combined with VAE. CRISP [17] applies critical path analysis for large-scale microservices. Seer [18] leverages large-scale tracing data for performance debugging. Additional methods such as MEPFL [19] and TraceVAE [20] further improve anomaly detection through representation learning and graph-based modeling.

Despite their effectiveness, single-modal approaches fail to capture correlations between heterogeneous data sources, leading to limited detection performance.

2.2 Multimodal Anomaly Detection

To tackle the constraints of single-modal methods, recent approaches incorporate multiple data modalities.DeepTraLog [21] integrates logs and traces through graph neural networks. Hades [22] introduces cross-modal attention with shared latent representations. SCWarn [23] models the temporal dependencies for identifying anomalous software changes. Eadro [24] constructs dependency graphs to jointly model logs, metrics, and traces. AnoFusion [25] employs heterogeneous graph neural networks to learn the correlations among different modalities, but its graph structure does not reflect the spatiotemporal dependency of anomalies. DiagFusion [26] introduces graph neural networks to model inter-service relationships. MSTGAD [27] utilizes twin graph structures to capture multimodal interactions, and TraceGra [28] utilizes graph-based learning on trace data.

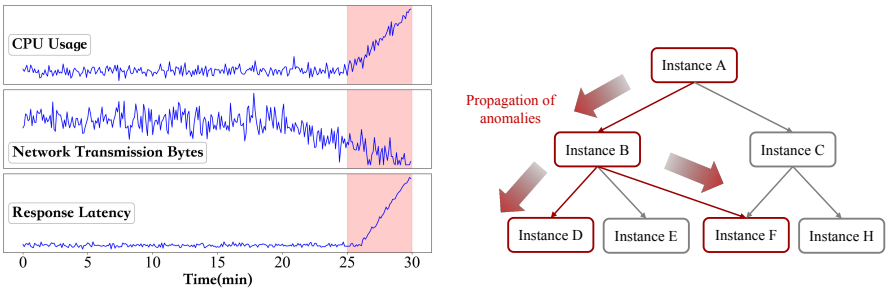
Although the above methods improve anomaly detection from different perspectives, such as cross-modal fusion, service dependency modeling, and temporal evolution modeling, they still fall short of comprehensively and jointly modeling multimodal correlations and spatiotemporal dependencies. Specifically, SCWarn and HAdes do not explicitly model service dependency relationships. SCWarn, MSTGAD, and DiagFusion mainly adopt loose fusion strategies or model different components separately, which may limit their ability to capture rich inter-modal correlations. DeepTraLog mainly focuses on service dependency modeling, whereas AnoFusion emphasizes inter-modal correlation modeling. Eadro connects temporal modeling and spatial modeling in a one-way sequential manner, which may not fully capture the interaction between temporal and spatial dependencies. Therefore, these methods still have limitations in spatiotemporal dependency modeling.

3 Challenges and Solutions

With the expansion of microservice systems and the increasing complexity of inter-service interactions, anomaly detection faces higher requirements. To ensure stable system operation and high-quality services, two major challenges in microservice anomaly detection must be addressed.

- Insufficient multimodal data fusion
  During the operation of microservice systems, large volumes of monitoring data are generated, exhibiting strong temporal continuity. Anomalies are rarely manifested as isolated deviations in a single modality or at a specific timestamp. Instead, they typically appear as coordinated trend changes across multiple modalities over a period of time, as illustrated in Figure 2(a). Therefore, effectively exploiting multimodal correlations is critical for improving anomaly detection accuracy.
- Lack of spatiotemporal dependency modeling
  Existing anomaly detection methods typically focus on either temporal dynamics or trace structures, while overlooking their joint effects. In microservice systems, anomalies are influenced not only by service dependency relationships but also by evolving operational states. As shown in Figure 2(b), anomalies tend to propagate

along service invocation traces. During this process, multimodal data exhibit consistent temporal patterns, and strong interactions exist between temporal dynamics and invocation dependencies. Modeling temporal information and invocation dependencies separately limits the model's ability to capture complex anomaly patterns. Therefore, spatiotemporal dependency modeling is essential for improving anomaly detection performance.



(a) Anomalous trends over time. (b) Anomalous trends along the call chain.

Fig. 2. Anomalous trends in temporal and call-chain dimensions.

To describe the research problem more clearly, this paper formally defines the microservice anomaly detection task. Assume that in a distributed system composed of N microservice instances, logs, metrics, and traces are collected separately for each microservice instance. Given a timestamp t, the multimodal data stream of the microservice system can be defined as $x_t = (x_t^{(metric)}, x_t^{(log)}, x_t^{(trace)})$, where $x_t^{(metric)}, x_t^{(log)}, x_t^{(trace)}$ stand for the metric, log, and trace data collected at time t s, respectively. The goal is to compute the anomaly probability P_t^{ano} based on the multimodal data stream x_t and the service dependency graph G, and then perform two-class classification aimed at determining whether the system is anomalous.

4 Methodology

The architecture of the STMID model is illustrated in Figure 3. The proposed framework performs instance-level anomaly detection and consists of three main components: data preprocessing, cross-modal fusion, and anomaly detection via spatiotemporal dependency modeling. In the data preprocessing stage, features are extracted from metrics, logs, and distributed traces. A graph structure is then constructed to model the relationships among data sources, enabling data fusion along both the temporal and the trace dimensions. In this graph, service instances are represented as nodes, while scheduling or invocation relationships between instances are modeled as edges. Cross-modal fusion learns a unified multimodal representation. Using an attention mechanism over a shared global fusion representation, cross-modal fusion captures global cross-modal context, facilitating effective information interaction and feature enhancement across

different modalities. Anomaly detection is conducted based on spatiotemporal dependency modeling. Both the encoder and decoder capture temporal correlations and invocation dependencies within the system. Specifically, a Graph Attention Network (GAT) is used to enhance the model's ability to learn structural features from the graph, while a Temporal Convolutional Network (TCN) is introduced to capture temporal dependencies among data points within a sliding window. Stacking multiple layers of GAT and TCN enables the model to learn higher-order spatiotemporal interactions. Similar to the encoder, the decoder also incorporates a GAT. In addition, a causal Temporal Convolutional Network (masked TCN) is adopted to ensure that the output at each time step depends only on current and past information, thereby preventing the use of future data and avoiding information leakage. The encoder output and the representations produced by the causal TCN are fused through an encoder–decoder attention module (EDCM), and the resulting latent representation is mapped into the latent space of a variational autoencoder (VAE). Finally, samples drawn from the latent space are fed into the decoder for data reconstruction, and anomalies are identified based on reconstruction errors.

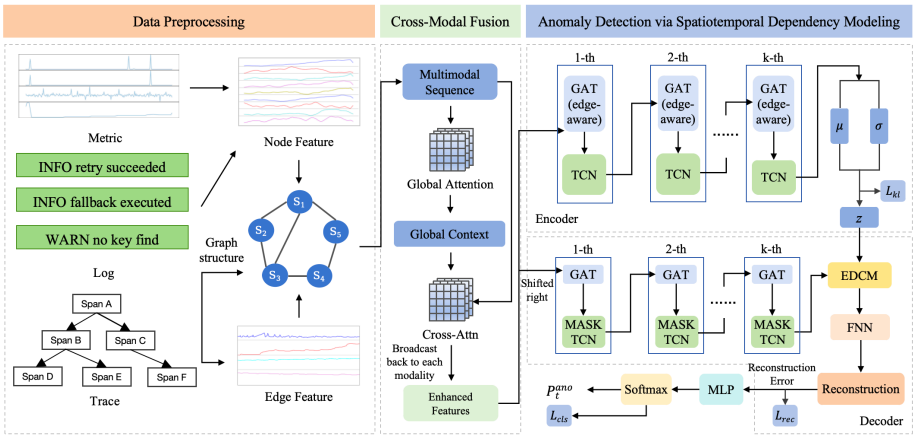


Fig. 3. The framework of STMID.

4.1 Data Preprocessing

In distributed microservice systems, logs, metrics, and traces are three core modalities of operational data. Together, they comprehensively record system states and events, reflect the status of services and hosts, and trace the execution paths of user requests. To enable effective anomaly detection, these unstructured or semi-structured data sources must be transformed into structured representations and integrated across modalities, thereby constructing a unified analytical framework.

Metrics capture system resource utilization and service performance characteristics, such as response time, thread count, and memory usage. In this work, we focus on multivariate time series data, denoted as $\Gamma_{Metric} = \{x_1^{(metric)}, x_2^{(metric)}, \ldots, x_T^{(metric)}\}$,

which represents a series of measurements over T timestamps. At each timestamp$t$, a data point $x_t^{(metric)} \in \mathbb{R}^{N \times F_m}$ is collected for N service instances, where F_m denotes the number of metric features. We further normalize the data and transform it into time-windowed samples suitable for training and evaluation. Each data point $x_t^{(metric)}$ is standardized by min–max normalization:

$$x_t^{(metric)} = \frac{x_t^{(metric)} - \min(\Gamma_{Metric})}{\max(\Gamma_{Metric}) - \min(\Gamma_{Metric})} \tag{1}$$

Here, max $(\Gamma_{Metric}) \in \mathbb{R}^{F_m}$ and min $(\Gamma_{Metric}) \in \mathbb{R}^{F_m}$ denote the maximum and minimum values, respectively, of each metric dimension within the training time series.

Logs are typically unstructured text, recording events and state changes from the operating system, applications, and security components, including log levels, message contents, and timestamps. For anomaly detection, massive log data must be processed and converted into a structured format to facilitate subsequent feature extraction and analysis. To this end, we employ the efficient streaming log parsing tool Drain3, which can rapidly process large-scale logs while maintaining high parsing accuracy. The parsed logs are organized into structured information, including timestamps, log levels, and log template indices, enabling unified system analysis and anomaly pattern identification.

To integrate logs with metrics for joint analysis, We record the frequency of each log template at every timestamp, Constructing a log representation aligned with the metric definition. This produces a timestamped log sequence, denoted as $\Gamma_{Log} = \{x_1^{(log)}, x_2^{(log)}, \ldots, x_T^{(log)}\}$, where each data point $L_t \in \mathbb{R}^{N \times F_l}$, with F_l representing the number of log templates. Analogous to metrics, each log data point L_t is normalized using min–max scaling:

$$x_t^{(log)} = \frac{x_t^{(log)} - min(\Gamma_{Log})}{max(\Gamma_{Log}) - min(\Gamma_{Log}) + \epsilon} \tag{2}$$

where max $(\Gamma_{Log}) \in \mathbb{R}^{Fl}$ and min $(\Gamma_{Log}) \in \mathbb{R}^{Fl}$ are the maximum and minimum values of each log template in the training time series, and $\epsilon = 10^{-6}$ is added to avoid division by zero.

Traces describe the execution paths of user requests within a microservice system, reflecting call chains and scheduling relationships among service instances. We aggregate trace spans into time series by computing the total duration of spans with identical request types and service instances and applying normalization. In a microservice system, each trace represents the execution of a user request, composed of multiple spans. Each span contains valuable attributes, such as trace ID, request type, span ID, parent span ID, service instance ID, start time, and duration. Spans are represented as time series, and within a sliding window, we aggregate the total duration of spans with the same request type, sender, and receiver service instances to form a timestamped sequence. Incomplete spans are excluded from the window. The resulting timestamped trace sequence is denoted as $\Gamma_{Trace} = \left\{x_1^{(trace)}, x_2^{(trace)}, \ldots, x_T^{(trace)}\right\}$, where each time

step $x_t^{(trace)} \in \mathbb{R}^{N\times N\times F_s}$, N is the number of service instances, and F_s denotes the feature dimension. Given that not all service instances communicate with each other, we use an approximate normalization approach to separate small spans from absent ones.

$$x_t^{(trace)} = \frac{x_t^{(trace)}}{\text{mean}(\Gamma_{Trace})} \tag{3}$$

where $\text{mean}(\Gamma_{Trace}) \in \mathbb{R}^{F_S}$ is the average of all span features within the sliding window.

After processing all modalities, we construct the service dependency graph. In a system with N microservice instances, the operational status of each instance and its scheduling relationships are represented by a service dependency graph $G = (V, A, E)$, where V is the set of nodes; E represents edge features, with each element e_{ij} defined as the cumulative invocation latency from instance i to j within the sliding window; and A represents node dependencies, where $a_{ij} = 1$ if a scheduling relationship exists between service instances i and j, and $a_{ij} = 0$ otherwise. Although service dependency graph is static, it captures the predominant interactions among services and can detect deviations from typical patterns for anomaly detection.

4.2 Cross-Modal Fusion

To fully exploit the latent correlations among multimodal data, we design a cross-modal fusion module. At time step t, the input features of each modality are denoted as $x_t^{(m)}$, where $m \in \{metric, log, trace\}$. Since different modalities may have heterogeneous feature dimensions, modality-specific linear projection functions are employed to map them into a shared latent space:

$$h_t^{(m)} = W_m x_t^{(m)} + b_m, m \in \{metric, log, trace\} \tag{4}$$

where W_m and b_m are learnable parameters associated with modality m.
The feature representations of different modalities at the same time step are concatenated to construct a unified cross-modal sequence representation:

$$S = Concat_{seq}\left(h_t^{(metric)}, h_t^{(log)}, h_t^{(trace)}\right) \tag{5}$$

where $Concat_{seq}(\cdot)$ denotes concatenation along the sequence dimension.

Next, a global fusion representation tk_0 is introduced to aggregate global contextual information from the cross-modal sequence S. Compared with directly modeling pairwise attention across modalities, this design enables more efficient cross-modal interaction. The global fusion token is jointly optimized with other network parameters during training. Specifically, tk_0 is used as the query, while S serves as both keys and values. A multi-head attention mechanism $MHA(\cdot)$ is then applied to obtain the global context representation tk:

$$tk = MHA(Q = tk_0, K = S, V = S) \tag{6}$$

Each modality subsequently queries the global cross-modal context and incorporates it into its own feature representation. Concretely, the global context tk is broadcast back to each modality, where the modality-specific representation serves as the query, and tk acts as both keys and values:

$$X^{(m)} = LN\left(x^{(m)} + Proj^{(m)}\left(MHA\left(S^{(m)}, tk, tk\right)\right)\right) \tag{7}$$

Here, $S^{(m)}$ and $X^{(m)}$ denote the modality-specific sub-sequence extracted from S and the resulting enhanced representation, respectively. $Proj^{(m)}$ is a linear projection that map features back to the original dimensionality, and LN denotes Layer Normalization.

Through this cross-modal fusion mechanism, the model effectively captures inter-modal dependencies and enhances its capability to identify complex anomaly patterns in distributed systems.

4.3 Spatiotemporal Dependency Modeling

Encoder. To capture the correlations of system anomalies along service dependencies and temporal evolution, we design a spatiotemporal dependency encoder. The encoder constructs service call relations based on the service dependency graph and leverages temporal convolutions to model the time-evolving patterns of anomalies. Specifically, the encoder consists of a hierarchical structure of edge-aware GAT and TCN, with inputs being the multimodal representations $X^{(m)}$ enhanced by the cross-modal fusion module.

On the service dependency graph G, node features v_i are composed of the fused metric and log modality representations $X^{(metric)}$ and $X^{(log)}$, while edge features e_{ij} are derived from the fused trace modality representation $X^{(trace)}$. At each layer, the model first aggregates cross-modal fused node features via the edge-aware GAT to capture dependencies among service instances. The attention weights for edges are computed as:

$$w_{ij} = softmax_{j \in N(i)}\left(a^T ReLU\left[W_s v_i \,\middle\|\, W v_j \,\middle\|\, W_e e_{ij}\right]\right) \tag{8}$$

where w_{ij} is the attention weight of node i on its neighbor j, v_i is the input feature of node i, a^T, W, W_s and W_e are learnable parameters, and the softmax normalization is over the neighbor set $N(i)$.

After edge-aware GAT, the node embedding is $v_i' = ReLU(\sum_{j \in N(i)} w_{ij} W v_j)$. TCN captures long-range temporal dependencies, producing the temporally enhanced representation:

$$z_i(t) = \sum_{k=0}^{K-1} W_k \, v_i'(t-k) \tag{9}$$

where $z_i(t)$ is the TCN output, K is the temporal kernel size, and W_k are learnable parameters. Multiple layers of edge-aware GAT and TCN can be stacked to capture high-order temporal and trace features.

Variational Inference. To model the latent distribution of system states and enhance generalization to anomalies, we introduce variational inference. For each modality, the encoder outputs are mapped to the mean and variance of a latent distribution, followed by sampling via the reparameterization trick to obtain latent variables. The variational distribution for each modality is:

$$q\left(Z_t^{(m)} \middle| E_t^{(m)}\right) = N\left(\mu^{(m)}\left(E_t^{(m)}\right), \sigma^{2(m)}\left(E_t^{(m)}\right)\right) \tag{10}$$

where $\mu^{(m)}$ and $\sigma^{2(m)}$ are predicted by the encoder to parameterize the approximate posterior. $E_t^{(m)}$ denotes the encoder output of modality m at time step t.

Decoder. To model the current system state using historical information and improve anomaly discrimination, the decoder reconstructs system states from the latent representations $Z_t^{(m)}$. An Encoder–Decoder Cross-Attention Module (EDCM) based on multi-head attention dynamically focuses on the most relevant historical features. Its output is:

$$H_t^{(m)} = MHA\left(Q = d_t^{(m)}, K = Z_t^{(m)}, V = Z_t^{(m)}\right) \tag{11}$$

where $d_t^{(m)}$ is the decoder input for modality m, obtained by embedding the cross-modal fused sequence $X^{(m)}$ with a temporal shift.

The decoder then generates the reconstructed modality:

$$\hat{x}_t^{(m)} = f_{dec}\left(H_t^{(m)}\right) \tag{12}$$

Next, the reconstruction error $r_t^{(m)}$ for modality m at time step t is computed based on the real input $x_t^{(m)}$ and the reconstructed output $\hat{x}_t^{(m)}$.

$$r_t^{(m)} = \frac{1}{2}\left(x_t^{(m)} - \hat{x}_t^{(m)}\right)^2 \tag{13}$$

The modality-level errors are concatenated to form a system-level error representation r_t, capturing the deviation of the current system state from normal behavior.

$$r_t = Concat\left(r_t^{(metric)}, r_t^{(log)}, r_t^{(edge)}\right) \tag{14}$$

Finally, to perform anomaly discrimination, the system-level error representation r_t is fed into a Softmax function to generate binary classification probabilities:

$$P_t = Softmax(W_o r_t + b_o) \tag{15}$$

where $P_t = [P_t^{nor}, P_t^{ano}]$ denotes the predicted probabilities that the system is in a normal or anomalous state at time step t, respectively.

Through simultaneous modeling the reconstruction process in conjunction with discrimination process, the decoder is able not only to learn the latent distribution of normal system operations, but also to enhance its capability to distinguish anomalous patterns.

4.4 Spatiotemporal Dependency Modeling

Based on the variational encoder–decoder framework constructed in Section 4.3, the model is trained end-to-end using a joint optimization strategy. The prior distribution of the latent variable is assumed to follow a standard Gaussian distribution, $p(z) = \mathcal{N}(0, I)$, which enables the Kullback–Leibler (KL) divergence to be expressed in closed form. By integrating the variational reconstruction objective with the discriminative classification objective, the overall training objective can be formulated as follows:

$$L = L_{rec} + \beta L_{kl} + \lambda L_{cls} \tag{16}$$

$$L_{rec} = \sum_{m} r_t^{(m)} \tag{17}$$

$$L_{kl} = \frac{1}{M} \sum_{m} D_{KL}(q(Z_t^{(m)} | E_t^{(m)}) || p(Z_t^{(m)})) \tag{18}$$

$$L_{cls} = -(y log P_T^{ano} + (1 - y) log P_T^{nor}) \tag{19}$$

where L_{rec} denotes the reconstruction loss, L_{kl} represents the KL regularization term, and L_{cls} is the cross-entropy classification loss. The coefficients β and λ are weighting factors for the corresponding loss terms. P_T^{nor} and P_T^{ano} denote the predicted probabilities that the model is in the normal or anomalous state at the final time step, respectively, and y represents the ground-truth label. Since the expectation term in the reconstruction objective is difficult to compute directly, the reparameterization trick is adopted to sample the latent variables, and the expectation is approximated using Monte Carlo estimation.

5 Experiment

We evaluate the proposed method through the following research questions.

- RQ1: How well does STMID perform in microservice system
- anomaly detection?
- RQ2: Does each component contribute to STMID?
- RQ3: How do the major parameters of STMID influence its performance?

5.1 Experimental Design

Experimental Setup. All experiments were performed on a server equipped with the following hardware setup: a 12-core CPU, 85 GB RAM, and an NVIDIA A100 GPU with 40 GB of memory. The operating system used was Ubuntu 20.04 LTS. The experimental environment included Python 3.10, PyTorch 2.1, and CUDA 11.8.

Through hyperparameter tuning, the encoder and decoder were set to 2 layers, the batch size was 50, the sliding window size was 10, and dropout was set to 0.2. The AdaBelief optimizer was used with an initial learning rate of 0.001 and a step-based learning rate scheduler with a decay factor of 0.9. The model was trained for a maximum of 300 epochs, and early stopping with a patience of 15 was applied.

Datasets. To evaluate the performance of the model, experiments were conducted on two widely used open-source multimodal microservice datasets: MSDS [29] and GAIA [30]. Table 1 lists the detailed information of the two datasets.

- D1 is the Multi-Source Distributed System (MSDS) dataset which contains traces, logs, and metrics collected from a complex distributed system used to support AI-driven analytical tasks. Metric data were collected from five physical nodes in the infrastructure, with each node monitored for seven metrics, including memory and CPU usage. Log data were distributed across the system nodes and included 23 features. Trace data contained multiple attributes, such as hostname, event name, service name, Span ID, parent Span ID, and Trace ID. The dataset covers approximately 28 hours of system operation with a sampling frequency of 1 second.
- D2 is the Generic AIOps Atlas (GAIA) dataset collected from the MicroSS business simulation system provided by Cloudwise, including metrics, logs, and traces. By controlling user behavior and simulating erroneous operations, GAIA captures potential system anomaly injection processes, with complete injection logs provided for algorithm evaluation. The system consists of five hosts, ten microservices, and two database services (MySQL and Redis). GAIA continuously collects system operation data for approximately two weeks, with a sampling frequency of 30 seconds.

Table 1. Dataset statistics.

Dataset	Instances	Anomaly Ratio	Modality	Data Size
MSDS	5	0.78%	Metrics	11,157
			Logs	130,326
			Traces	1,514,358
GAIA	40	4.91%	Metrics	117,411,233
			Logs	80,113,843
			Traces	26,064,740

For all datasets, 60% of the data was used for training, 10% for validation, and the remaining 30% for testing. Since the proposed STMID method is a semi-supervised approach, 50% of the training data was randomly selected as unlabeled. During training, the labeled data was used for supervised classification and reconstruction learning,

while the unlabeled data participated only in reconstruction loss optimization and was excluded from classification loss calculation. This design avoids full reliance on anomaly labels and simulates scenarios with limited labeled anomalies.

Performance Metrics. We adopt widely used performance measures, including Precision (PR), Recall (RC), and F1-score ($F1$) to evaluate anomaly detection performance. PR and RC measure the accuracy and anomaly coverage of the model, respectively, while $F1$, defined as the harmonic mean of PR and RC, evaluates overall detection capability. The metrics are computed as:

$$PR = \frac{TP}{TP + FP} \tag{20}$$

$$RC = \frac{TP}{TP + FN} \tag{21}$$

$$F1 = \frac{2 \times PR \times RC}{PR + RC} \tag{22}$$

where TP (true positives) denotes anomalies successfully detected, FP (false positives) denotes benign instances misidentified as anomalous, and FN (false negatives) denotes anomalies missed by the model.

Baselines. We compare STMID with five baseline methods: single-modality anomaly detection approaches TranAD [3], TraceAnomaly [15], and DeepLog [9], and multimodal anomaly detection approaches SCWarn [23] and AnoFusion [25]. The principles of these baseline models are introduced in the related work, and their utilized data modalities and technical details are summarized in Table 2.

Table 2. Techniques and data modalities of baseline methods.

Method	Main Technique	Data Modality
TranAD	Transformer + Adversarial Training	Metric
TraceAnomaly	Deep Bayesian Network	Trace
LogAnomaly	Log Key Sequence + LSTM	Log
SCWarn	LSTM + Multimodal Feature Fusion	Metric, Log
AnoFusion	Graph Transformer + Graph Attention Network	Metric, Log, Trace

5.2 RQ1: Overall Performance of STMID

This study aims to evaluate the overall performance of STMID in anomaly detection and to compare it with five representative anomaly detection methods, including TranAD, TraceAnomaly, DeepLog, SCWarn and AnoFusion, covering the mainstream

approaches in the field. The results are shown in Table 3. On the two datasets D1 and D2, STMID achieves a Precision of 0.939 and an F1-score of 0.885 on D1, and a Precision of 0.922 and an F1-score of 0.873 on D2, outperforming most of the comparison methods overall. Compared with the single-modal methods TranAD and DeepLog, STMID improves detection precision. Compared with the multimodal methods SCWarn and AnoFusion, STMID achieves the best results on most metrics. SCWarn obtains an F1-score of only 0.407 on D1 and 0.492 on D2, with relatively low performance, likely because it ignores interactions between service instances. AnoFusion achieves the best Recall on D2, but its F1-score is still lower than STMID. AnoFusion performs anomaly detection separately on each modality and does not fully model cross-modal correlations, which could limit its performance. It should be noted that STMID has a Recall of 0.837 on D1, which is lower than TraceAnomaly's 0.896, and a Recall of 0.829 on D2, slightly below AnoFusion's 0.905, indicating that STMID is more conservative in detecting some anomalies while pursuing high Precision. Overall, STMID achieves high precision and competitive F1-scores on both datasets, demonstrating its effectiveness in anomaly detection tasks.

Table 3. Performance comparison of baseline methods.

Method	D1			D2		
	Precision	Recall	F1-score	Precision	Recall	F1-score
TranAD	0.776	0.806	0.790	0.754	0.760	0.757
TraceAnomaly	0.907	**0.896**	**0.901**	0.916	0.824	0.867
DeepLog	0.816	0.760	0.787	0.798	0.762	0.780
SCWarn	0.438	0.380	0.407	0.481	0.504	0.492
AnoFusion	0.838	0.819	0.828	0.808	**0.905**	0.853
STMID	**0.939**	0.837	0.885	**0.922**	0.829	**0.873**

5.3 RQ2: Contributions of Components

To evaluate the contribution of each module in STMID to anomaly detection performance, we designed four ablation experiments. STMID-CMF was run without the cross-modal fusion module to assess the importance of integrating information across modalities. STMID-GAT was run without the graph attention module while keeping all other modules to analyze the effect of modeling service instance relationships. STMID-TCN was run without the temporal convolutional network module to examine the contribution of temporal feature modeling. The full STMID retained all modules and served as the performance baseline.

The results in Table 4 show that STMID-CMF achieved F1-scores of 0.866 on D1 and 0.813 on D2, both lower than those of the full STMID. STMID-GAT and STMID-TCN obtained F1-scores of 0.868 and 0.865 on D1, and 0.837 and 0.845 on D2, respec-

tively, which are also below the full STMID, indicating that removing any module negatively impacts model performance. Analysis suggests that the performance drop of STMID-CMF is mainly due to the absence of the cross-modal fusion module, which prevents complementary information across modalities from being fully utilized. The performance decrease of STMID-GAT indicates that modeling service instance relationships is crucial for capturing system anomalies, while the decline of STMID-TCN shows that temporal modeling is essential for recognizing anomaly patterns. Ultimately, the full STMID achieved F1-scores of 0.885 on D1 and 0.873 on D2, outperforming all ablation configurations. The results suggest that the integration of cross-modal fusion, graph attention, and temporal modeling modules positively impacts the model's overall performance.

Table 4. Ablation study results of STMID modules.

Method	D1			D2		
	Precision	Recall	F1-score	Precision	Recall	F1-score
STMID-CMF	0.867	**0.866**	0.866	0.836	0.791	0.813
STMID-GAT	0.929	0.814	0.868	0.880	0.798	0.837
STMID-TCN	0.886	0.845	0.865	0.869	0.822	0.845
STMID	**0.939**	0.837	**0.885**	**0.922**	**0.829**	**0.873**

5.4 RQ3: Parameter Sensitivity

To investigate how principal hyperparameters affect STMID on anomaly detection performance, we conducted a parameter sensitivity study on the MSDS dataset, focusing on sliding window size, TCN convolutional layers, GAT attention heads, and the depth of encoder-decoder.

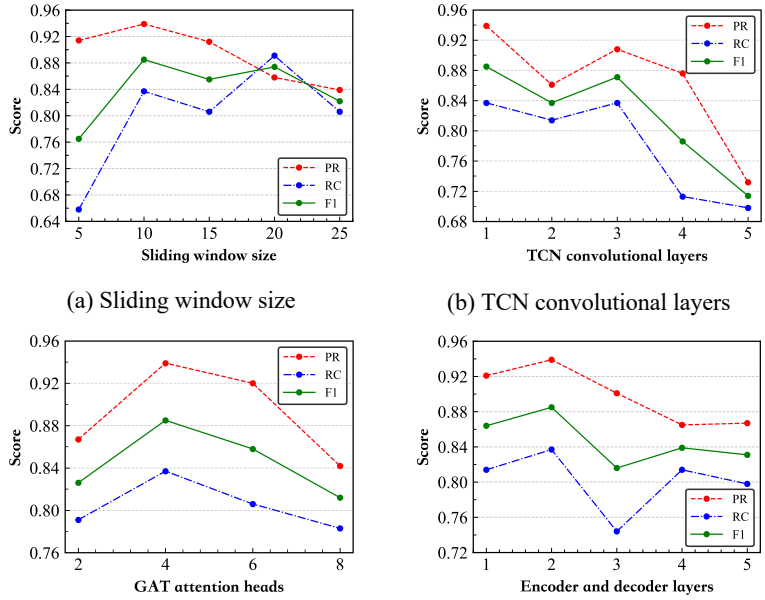
Figure 4(a) illustrates the effect of sliding window size on model performance. When the window size is 5, the temporal context captured by the model is limited, resulting in a relatively low Recall. Increasing the window to 10 substantially improves both Precision and Recall, yielding the highest F1-score. Further increasing the window size to 20 or 25 causes performance to decline. These results indicate that a window that is too small fails to capture essential temporal features, whereas an excessively large window introduces irrelevant information and noise.

Figure 4(b) shows the influence of the number of TCN convolutional layers. A single TCN layer achieves the highest F1-score, while increasing to 2 or 3 layers results in a slight decrease. When the number of layers exceeds 3, F1-score drops more noticeably. This phenomenon suggests that too few layers restrict the model's capacity to learn long-term temporal patterns, whereas too many layers lead to over-smoothing, reducing the discriminability of anomaly patterns.

Figure 4(c) presents the impact of the GAT attention heads. With 2 attention heads, F1-score is relatively low; increasing to 4 heads produces peak performance. Further increasing the number of heads to 6 or 8 leads to a decline in F1-score. This indicates

that too few attention heads constrain the integration of node features in the graph structure, while too many heads disperse information and degrade performance.

Figure 4(d) depicts the effect of the encoder-decoder depth. Two layers achieve the highest F1-score, whereas a single layer reduces performance. Increasing the depth to 3, 4, or 5 layers results in a gradual decrease in F1-score. These observations suggest that too few layers reduce the model's effectiveness in learning long-range temporal patterns and multi-layer invocation relationships, while excessive layers introduce over-smoothing, causing anomaly features to become increasingly similar across layers.



(a) Sliding window size (b) TCN convolutional layers

(c) GAT attention head (d) Encoder–decoder layers

Fig. 4. Parameter sensitivity analysis of STMID on the MSDS dataset.

6 Conclusion

In this paper, we proposed STMID, a spatiotemporal multimodal interaction method for anomaly detection in microservice systems. STMID addresses the issues of insufficient multimodal data fusion and lack of spatiotemporal dependency modeling by incorporating cross-modal fusion and spatiotemporal dependency modeling. Experimental results demonstrate that STMID outperforms most baseline methods, with improvements ranging from 3.85% to 42.95%. Notably, STMID achieves high precision and competitive F1-scores, indicating its capability in performing anomaly detection. Since this work primarily focuses on algorithm development and evaluation, real-time deployment, including inference latency and computational overhead, will be considered in future work.

References

1. Soldani, J., Tamburri, D.A., Van Den Heuvel, W.-J.: The pains and gains of micro-services: A systematic grey literature review. Journal of Systems and Software 146, 215–232 (2018)
2. Zimmermann, O.: Microservices tenets: Agile approach to service development and deployment. Computer Science - Research and Development 32(3), 301–310 (2017)
3. Tuli, S., Casale, G., Jennings, N.R.: TranAD: Deep transformer networks for anomaly detection in multivariate time series data. arXiv:2201.07284 (2022)
4. Su, Y., Zhao, Y., Niu, C., et al.: Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In: KDD, pp. 2828–2837 (2019)
5. Audibert, J., Michiardi, P., Guyard, F., et al.: USAD: Unsupervised anomaly detection on multivariate time series. In: KDD, pp. 3395–3404 (2020)
6. Li, D., Chen, D., Jin, B., et al.: MAD-GAN: Multivariate anomaly detection for time-series data with generative adversarial networks. In: ICANN, pp. 703–716 (2019)
7. Zhao, H., Wang, Y., Duan, J., et al.: Multivariate time-series anomaly detection via graph attention network. In: ICDM, pp. 841–850 (2020)
8. Shi, Y., Wang, B., Yu, Y., et al.: Robust anomaly detection for multivariate time series through temporal GCNs and attention-based VAE. Knowledge-Based Systems 275, 110725 (2023)
9. Du, M., Li, F., Zheng, G., et al.: DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In: CCS, pp. 1285–1298 (2017)
10. Meng, W., Liu, Y., Zhu, Y., et al.: LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In: IJCAI, pp. 4739–4745 (2019)
11. Zhang, X., Xu, Y., Lin, Q., et al.: Robust log-based anomaly detection on unstable log data. In: ESEC/FSE, pp. 807–817 (2019)
12. Fu, Y., Liang, K., Xu, J.: MLog: Mogrifier LSTM-based log anomaly detection approach using semantic representation. IEEE Transactions on Services Computing 16(5), 3537–3549 (2023)
13. Xie, Y., Yang, K.: Log anomaly detection by adversarial autoencoders with graph feature fusion. IEEE Transactions on Reliability 73(1), 637–649 (2023)
14. Yang, L., Chen, J., Wang, Z., et al.: PLELog: Semi-supervised log-based anomaly detection via probabilistic label estimation. In: ICSE Companion, pp. 230–231 (2021)
15. Liu, P., Xu, H., Ouyang, Q., et al.: Unsupervised detection of microservice trace anomalies through service-level deep Bayesian networks. In: ISSRE, pp. 48–58 (2020)
16. Cai, Y., Han, B., Su, J., et al.: TraceModel: An automatic anomaly detection and root cause localization framework for microservice systems. In: MSN, pp. 512–519 (2021)
17. Zhang, Z., Ramanathan, M.K., Raj, P., et al.: CRISP: Critical path analysis of large-scale microservice architectures. In: USENIX ATC, pp. 655–672 (2022)
18. Gan, Y., Zhang, Y., Hu, K., et al.: Seer: Leveraging big data to navigate the complexity of performance debugging in cloud microservices. In: ASPLOS, pp. 19–33 (2019)
19. Zhou, X., Peng, X., Xie, T., et al.: Latent error prediction and fault localization for microservice applications by learning from system trace logs. In: ESEC/FSE, pp. 683–694 (2019)
20. Xie, Z., Xu, H., Chen, W., et al.: Unsupervised anomaly detection on microservice traces through graph VAE. In: WWW, pp. 2874–2884 (2023)
21. Zhang, C., Peng, X., Sha, C., et al.: DeepTraLog: Trace-log combined microservice anomaly detection through graph-based deep learning. In: ICSE, pp. 623–634 (2022)
22. Lee, C., Yang, T., Chen, Z., et al.: Heterogeneous anomaly detection for software systems via semi-supervised cross-modal attention. In: ICSE, pp. 1724–1736 (2023)

23. Zhao, N., Chen, J., Yu, Z., et al.: Identifying bad software changes via multimodal anomaly detection for online service systems. In: ESEC/FSE, pp. 527–539 (2021)
24. Lee, C., Yang, T., Chen, Z., et al.: Eadro: An end-to-end troubleshooting framework for microservices on multi-source data. In: ICSE, pp. 1750–1762 (2023)
25. Zhao, C., Ma, M., Zhong, Z., et al.: Robust multimodal failure detection for microservice systems. In: KDD, pp. 5639–5649 (2023)
26. Zhang, S., Jin, P., Lin,Z., et al.: Robust failure diagnosis of microservice system through multimodal data. IEEE Transactions on Services Computing 16(6), 3851–3864 (2023)
27. Huang, J., Yang, Y., Yu, H., et al.: Twin graph-based anomaly detection via attentive multi-modal learning for microservice system. In: ASE, pp. 66–78 (2023)
28. Chen, J., Liu, F., Jiang, J., et al.: TraceGra: A trace-based anomaly detection for micro-service using graph deep learning. Computer Communications 204, 109–117 (2023)
29. Nedelkoski, S., Bogatinovski, J., Mandapati, A.K., et al.: Multi-source distributed system data for AI-powered analytics. In: ESOCC, pp. 161–176 (2020)
30. CloudWise: GAIA: Generic AIOps Atlas dataset, https://github.com/CloudWise-Open-Source/GAIA-DataSet, last accessed 2025/10/25